Data Quality Assessment of Large-Scale Open Dataset

Tianxiang Chen, Yu-Lin Shen, Yuning Song

Github *Repository:* *https://github.com/chenttxx/BigDataProject*

I. Introduction

There could be various types of data quality issues in large-scale open dataset that would compromise the usability and reliability of the dataset. Data Quality Assessment is the process of applying guidelines and techniques to scientifically identify and determine the data issues, such that to improve the data quality to fulfill the use purpose. In this proposal, we use the big data framework, pyspark, to identify and analyze multiple most common data quality issues present in a large dataset from NYC Open Data.

II. Problem Formation

Null values, invalid format values, misspellings of valid non_numeric values and outliers are the most common types of data quality issues. We seek to identify these issues from the large-scale of the dataset and provide the corresponding solutions. The choice of the dataset is Office of Payroll Administration [1] from NYC Open Data.

III. Related Work

1. Null Values

Null Values are fields with missing values and the presence of null values could be due to data entry errors or data collection problems. It remains important to handle null values in the dataset if not set on purpose, because bias exists in statistical results based on the dataset containing a large number of non-random missing values. Also, many real-world models do not support null values in the dataset [3].

The handling of null values is the first basic step of cleaning the dataset. The process of handling null values in the dataset has been developed well in the Big Data field. It has been a rather standard procedure of first identifying the Null Values distribution in the dataset. Then if null values become a significant problem in a row or a column that it would compromise the usability of the dataset, relative columns or rows should be dropped with additional manual judgement. If null value percentage does not pass the tolerance threshold, no data would be dropped and replacements of null values with appropriate data could be implemented. Popular choices of replacing values for null fields include average value, mode value, median value, or a symbol for suggesting 'Missing'. [2]

2. Invalid Format

Inspired by Python Regex (Regular Expressions) for Data Scientists[22], we can use regex to match all items with the valid format in the column. Regular expressions (regex) are essentially text patterns that you can use to automate searching through and replacing elements within strings of text. This can make cleaning and working with text-based data sets much easier, saving you the trouble of having to search through mountains of text by hand.

Invalid format could be a simple but usual question for data cleaning. Unifying the column format might be a frequent technique on handling the dataset. To unify it, one of the common ways for dealing string data is using regex expressions. In [18], a regular expression provides a concise and flexible means to match (specify and recognize) strings of text, such as particular characters, words, or patterns of characters. Common abbreviations for "regular expression" include regex and regexp. Before unifying the format, checking the correct rate is also important. For those columns with 100% correct rate, we can ignore them. [17]

3. Misspellings of Non_Numeric Values

Misspelling on the text is one of the key issues in data mining. One of the methods to avoid the misspelling is finding the extra dataset as the correct value to make the further correction. If the outside dataset cannot be used, the frequency of the word in the original dataset will be used. But no matter what methods that we choose to find the misspelled value, computing the similarity between words, which means the distance, is necessary.

Before computing the distance, we need to transform the word to the vector first. One of the easiest ways to do is split the vocabulary into a set of letters and use one hot encoding on all the letters set to generate the vector. But this cannot preserve the order of the alphabet letter. So the N gram [14] technique has been adapted. Instead of just using one letter as the vector index, it will use n letters. For example, church, for 2 gram, (ch, hu, ur, rc, ch) will be generated; for 3 gram, (chu, hur, urc, rch) will be used. This helps to preserve some of the structure of the original vocabularies.

For distance computation on the vector, cosine similarity is the most common use [15]. It computes the cosine of the two vectors and provides the similarity. With different types of vectors that we used, the cosine similarity might be different. Another technique is levenshtein similarity [16], unlike the vector similarity, it just computes the letter difference between two vocabularies. The definition of the difference is to compute the times for insertion, deletion, replacement of a letter from one word to another word.

With the similarity, we can distinguish the misspelled words from the others and also make further corrections on those words.

4. Outlier

In order to find the outlier, several methods have been proposed. For example, interquartile range [8] might be the first thing that comes into our mind. For the interquartile range method, we first need to compute the range between quartile 25% (Q1) and quartile 75% (Q3). Then, if the data points are outside the range of (Q1 - 1.5*range, Q3 + 1.5*range), these points will be regarded as the outlier.

Another technique is using the clustering. K-means clustering will be the first thing that shows up in everyone’s mind. To run through it, we first need to define the cluster number K. The model will initial K center of the cluster and finds the distance between all the data points and these centers. After clustering, it will use the average method to compute the new centers in each cluster. This process will be one iteration. After running through several iterations, the centers will converge to K points and won't be modified a lot. With these clusters, we can define a specific distance range to pick the outliers.

Other clustering algorithms may show up frequently on the internet are Gaussian Mixture Model (GMM) [11] and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [10]. The GMM [12], basically, is the general version of k-means clustering. Rather than compute the closest point to the cluster (circle), it can handle different kinds of shape of the cluster, including spherical, tied, diag, etc. Although it is more flexible and can be implemented fast, the result is highly unstable, which is sensitive to the assumption, however [13]. The result for GMM might be different every time after implementation. For the DBSCAN, it is used to cluster the data point with the dense distribution. With this algorithm, it will combine the dense data points that are close to each other, which is a great advantage. But compared to the K-Means, the computation time is way longer, because it needs to find the relationship at each point.

IV. Methods, Architecture and Design

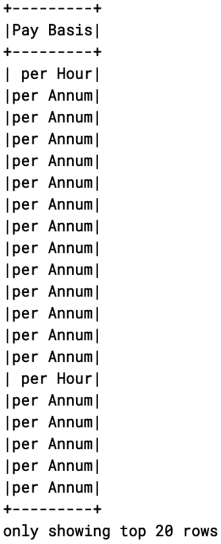
1. Null Values

We first evaluate the percentage of null values in each column before processing. If the percentage passes the threshold of tolerance, the whole column should be manually dropped from the dataset, if the column is considered to serve little informational significance to the dataset for the observer.

Then we handle null values with respect to numeric and non_numeric columns. For numeric columns, we replace missing values with the mean of the column, and for non_numeric columns, the mode value of the column should fill the missing value fields in the column.

2. Invalid Format

For invalid format, since different columns may have different values, we may need to set new formats for new columns. Here we simply set one column, Pay Basis, as our example. And look at the dataset, we can see this column has an obvious format: per xxx (xxx is a period).



```
+---------+
|Pay Basis|
+---------+
| per Hour|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
| per Hour|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
|per Annum|
+---------+
only showing top 20 rows
```

Pay Basis sample

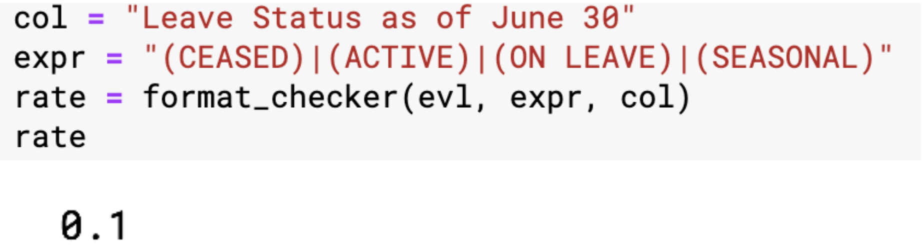
Before formatting, we need to check the correct rate for this column. We can use rlike function to be the checker:

rate = df.filter(df[col].rlike(expr)).count()/df.count() [19]

And the correct rate is 0.995, so we still need to format some irregular data.

To format the dataset, we need to check if the original data obeys the correct format. To realize this we need to use regex expressions. In this column, the expression is “^(per)\s((Hour)|(Annum)|(Day))”. And for different situations, we may decide to output various values. For example, for those strings containing “day”, we can output “per Day”, those containing “year”, we can output “per Annum”. And for exceptions, we simply return “per unrecognized period” to maintain the format.

Besides, in most columns like “Leave Status as of June 30”, they are perfectly formatted. By using regex expression "(CEASED)|(ACTIVE)|(ON LEAVE)|(SEASONAL)", we can use the checker before and get the correct rate for this column is already 1.



```
col = "Leave Status as of June 30"
expr = "(CEASED)|(ACTIVE)|(ON LEAVE)|(SEASONAL)"
rate = format_checker(evl, expr, col)
rate
```

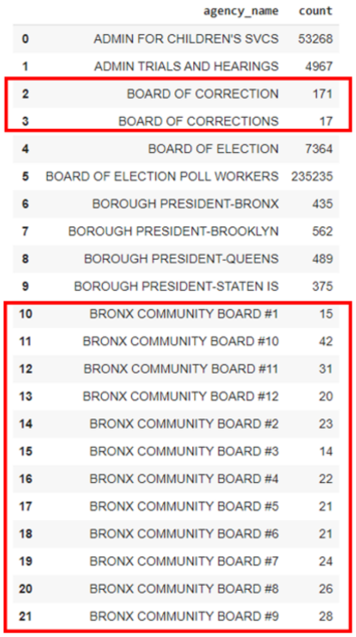
```
0.1
```

Using regex to match another column

3. Misspellings of Non_Numeric Values

For misspelling, we use the similarity and the frequency of words to find out the misspelling words and make further corrections.

In the “Agency Name” column, we first check the distribution of the column value. We can see the similar values, “BOARD OF CORRECTION” and “BOARD OF CORRECTIONS”, with different counts. The former is 171, and the latter is 17. Another example is the “BRONX COMMUNITY BOARD #” with different numbers from 1 to 12 at the end of the value. However, the distribution count among these are similar, which is 14 to 42. Obviously, some of the values are sure to be misspelling, but others might be depending on the definition. This means that, for example with the case “BRONX COMMUNITY BOARD #” with different numbers, you can regard the different number behind as the office number or the misspelling value that should be corrected.



	agency_name	count
0	ADMIN FOR CHILDREN'S SVCS	53268
1	ADMIN TRIALS AND HEARINGS	4967
2	BOARD OF CORRECTION	171
3	BOARD OF CORRECTIONS	17
4	BOARD OF ELECTION	7364
5	BOARD OF ELECTION POLL WORKERS	235235
6	BOROUGH PRESIDENT-BRONX	435
7	BOROUGH PRESIDENT-BROOKLYN	562
8	BOROUGH PRESIDENT-QUEENS	489
9	BOROUGH PRESIDENT-STATEN IS	375
10	BRONX COMMUNITY BOARD #1	15
11	BRONX COMMUNITY BOARD #10	42
12	BRONX COMMUNITY BOARD #11	31
13	BRONX COMMUNITY BOARD #12	20
14	BRONX COMMUNITY BOARD #2	23
15	BRONX COMMUNITY BOARD #3	14
16	BRONX COMMUNITY BOARD #4	22
17	BRONX COMMUNITY BOARD #5	21
18	BRONX COMMUNITY BOARD #6	21
19	BRONX COMMUNITY BOARD #7	24
20	BRONX COMMUNITY BOARD #8	26
21	BRONX COMMUNITY BOARD #9	28

The value distribution of column, Agency Name

To deal with this situation, this project presents two steps: cluster value with the similarity and distinguish the value with frequency. We will use the “Agency Name” as the sample column to state the method below.

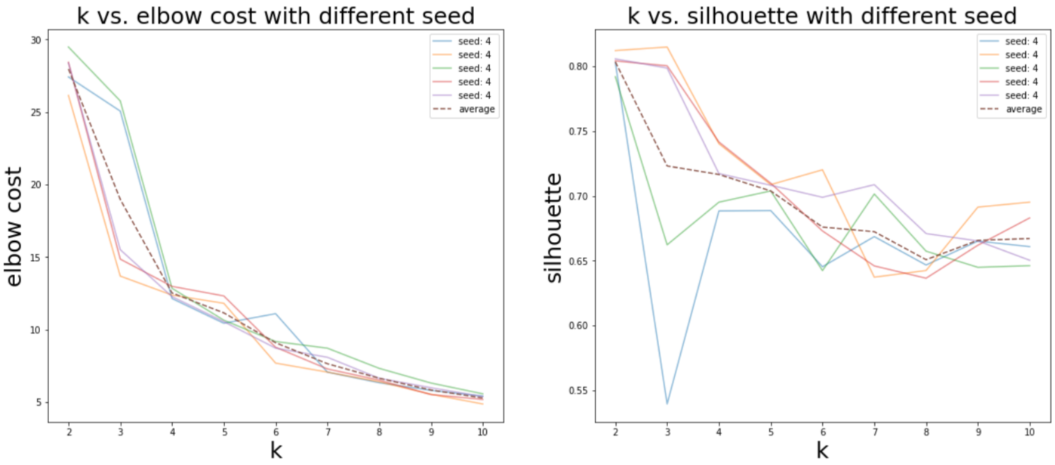
For similarity, we first gather the distinct value then use the levenshtein distance, [4], to compute the similarity among all the distinct values. Since our goal is to solve misspelling (assumption), we only search for the relation with levenshtein distance smaller than 2, which means giving 2 tolerance of letter misspelling. With the connection between words, we need to make the cluster. To elaborate, we can assume we have the record of relation words, say ((A, B), (B, C), (E, F), (F, G)). This means that A and B are similar, B and C are similar, etc. After that, we need to cluster the list to get the outcome, ((A, B, C), (E, F, G)). By doing this, we can join the similar words in each cluster. This can be implemented by aggregation function on DataFrame and mapping & reducing. Finally, we can get the different clusters with the similar word together.

With the similar word cluster, we now can apply the frequency count of the value to make the correctness on the misspelling. Originally, we set the threshold rate on the count difference in one cluster. If the high frequency is way larger than the low frequency, the highest frequency word might be the correct one; otherwise, the frequency with similar count as the sample has stated above will be further determined. However, setting the threshold is difficult and it will cause a huge unexpected effect on the result. Instead of computing the difference, we will check whether the letter difference in one cluster is alphabet or not. With the alphabet difference, we will pick the highest frequency count word as the correct word and regard other words as the misspelling. With the none-alphabet difference, such as numerical, we will only keep the same part in the alphabet and make it as the correct value. In this way, it is more intuitive and it also helps us to avoid the dilemma of choosing the threshold rate.

4. Outliers

For the outliers, we will focus on the numerical columns, such as Regular Salary, Pay Basis, and OT Hours. In general, we combine the method on interquartile range in statistics and K-Means. The method has three stages: first, we apply the K-Means on the data and, second, compute the distance between the center and the data points in different clusters. Finally, with these distances, we will compute the interquartile range in each cluster and search for the outlier.

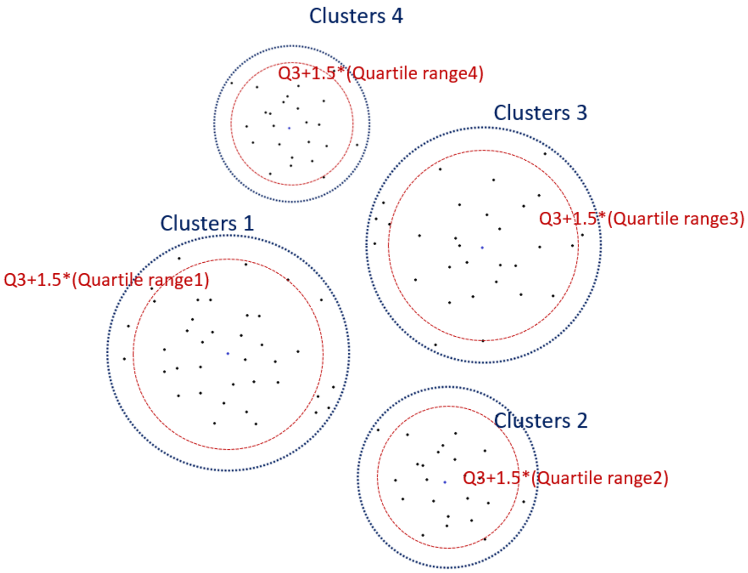
Before we apply the K-Means, we need to choose the optimal K on the cluster [7]. We use the Elbow Method and the Silhouette Method to pick K by sampling the data and run through the result fifth times. The result is shown below.



Elbow cost and silhouette method for picking k-means cluster k

Generally, for the elbow cost method, the point started to converge will be picked. With silhouette method, the highest will be choice. Here, the result is generated by the sampling dataset with different random seeds. We will choose the balance k, which is 4.

After picking the k, we normalized the data and started to run K-Means on pyspark.ml.cluster. Eventually, we get the center of each cluster and then compute the distance between the center point and the data point in each cluster. Usually, most of the project online will give the specific ratio or distance, such as [6] and [8], as the discriminator. If the distance difference between each cluster is larger than a specific number, the data point will be regarded as the outlier. But, still, picking the distance is a critical issue and hard to determine. Here, with all the distance we have from each data point to its center in different clusters, we computed the interquartile range of the distance in each cluster. If the distance is outside the interquartile range, the corresponding data point is the outlier.

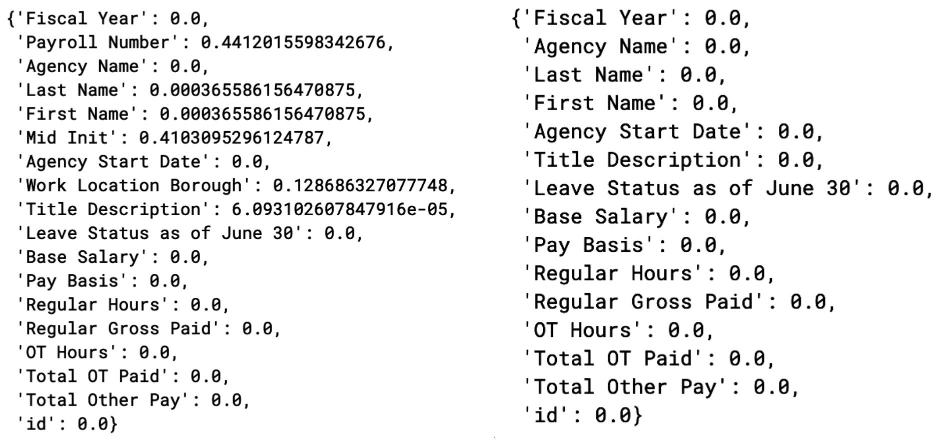


K-means cluster with quartile methods

V. Results & Evaluations

1. Null Values

With the replacements of null values in the dataset, the percentage of null value in each column becomes zero as below. Columns that are less important to the dataset with the significant amount of null values above the threshold are dropped.



```
{'Fiscal Year': 0.0,
 'Payroll Number': 0.4412015598342676,
 'Agency Name': 0.0,
 'Last Name': 0.000365586156470875,
 'First Name': 0.000365586156470875,
 'Mid Init': 0.4103095296124787,
 'Agency Start Date': 0.0,
 'Work Location Borough': 0.128686327077748,
 'Title Description': 6.093102607847916e-05,
 'Leave Status as of June 30': 0.0,
 'Base Salary': 0.0,
 'Pay Basis': 0.0,
 'Regular Hours': 0.0,
 'Regular Gross Paid': 0.0,
 'OT Hours': 0.0,
 'Total OT Paid': 0.0,
 'Total Other Pay': 0.0,
 'id': 0.0}
```

```
{'Fiscal Year': 0.0,
 'Agency Name': 0.0,
 'Last Name': 0.0,
 'First Name': 0.0,
 'Agency Start Date': 0.0,
 'Title Description': 0.0,
 'Leave Status as of June 30': 0.0,
 'Base Salary': 0.0,
 'Pay Basis': 0.0,
 'Regular Hours': 0.0,
 'Regular Gross Paid': 0.0,
 'OT Hours': 0.0,
 'Total OT Paid': 0.0,
 'Total Other Pay': 0.0,
 'id': 0.0}
```

Percentages of Null Values in each Column before and after Processing (1)

Columns	Percentage of Null Values before Processing	Percentage of Null Values after Processing
Payroll Number	0.441201559	Dropped
Last Name	0.000365586	0.0
First Name	0.000365586	0.0
Mid Init	0.410309529	Dropped
Work Location Borough	0.128686327	Dropped
Title Description	0.000006093	0.0

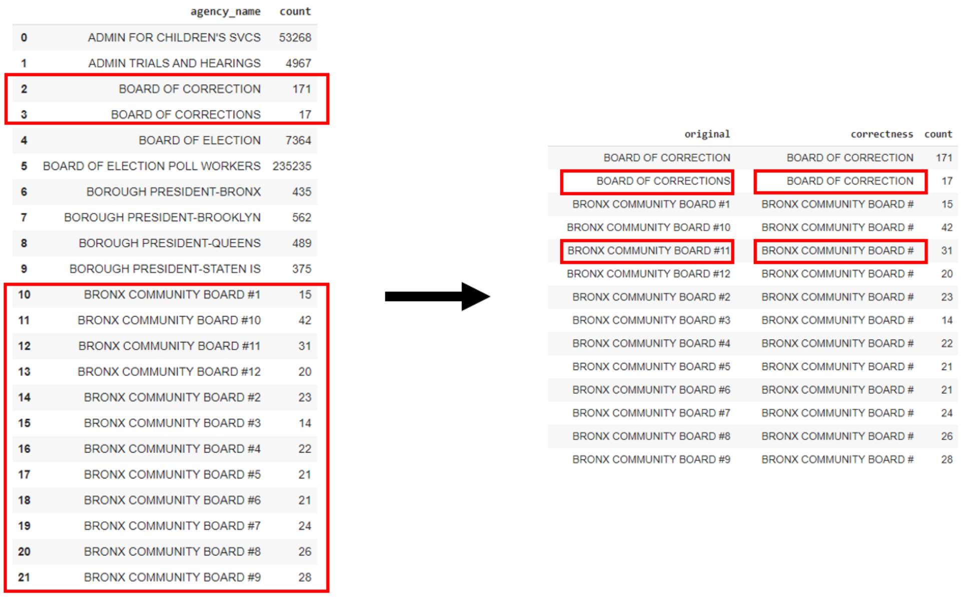
Null Value Processing Result Table (2)

2. Invalid Format

After unifying the format, all items in the column, such as Pay Basis, have the same format as below. The correct rate of those columns has increased from 0.995 to 1.

3. Misspellings of Non_Numeric Values

By running the program, we can get the result below, demonstrated by the “Agency Name” column. The misspelling value has been corrected and the similar value with different numbers has also been fixed by removing the decimal number.



	agency_name	count
0	ADMIN FOR CHILDREN'S SVCS	53268
1	ADMIN TRIALS AND HEARINGS	4967
2	BOARD OF CORRECTION	171
3	BOARD OF CORRECTIONS	17
4	BOARD OF ELECTION	7364
5	BOARD OF ELECTION POLL WORKERS	235235
6	BOROUGH PRESIDENT-BRONX	435
7	BOROUGH PRESIDENT-BROOKLYN	562
8	BOROUGH PRESIDENT-QUEENS	489
9	BOROUGH PRESIDENT-STATEN IS	375
10	BRONX COMMUNITY BOARD #1	15
11	BRONX COMMUNITY BOARD #10	42
12	BRONX COMMUNITY BOARD #11	31
13	BRONX COMMUNITY BOARD #12	20
14	BRONX COMMUNITY BOARD #2	23
15	BRONX COMMUNITY BOARD #3	14
16	BRONX COMMUNITY BOARD #4	22
17	BRONX COMMUNITY BOARD #5	21
18	BRONX COMMUNITY BOARD #6	21
19	BRONX COMMUNITY BOARD #7	24
20	BRONX COMMUNITY BOARD #8	26
21	BRONX COMMUNITY BOARD #9	28

original	correctness	count
BOARD OF CORRECTION	BOARD OF CORRECTION	171
BOARD OF CORRECTIONS	BOARD OF CORRECTION	17
BRONX COMMUNITY BOARD #1	BRONX COMMUNITY BOARD #	15
BRONX COMMUNITY BOARD #10	BRONX COMMUNITY BOARD #	42
BRONX COMMUNITY BOARD #11	BRONX COMMUNITY BOARD #	31
BRONX COMMUNITY BOARD #12	BRONX COMMUNITY BOARD #	20
BRONX COMMUNITY BOARD #2	BRONX COMMUNITY BOARD #	23
BRONX COMMUNITY BOARD #3	BRONX COMMUNITY BOARD #	14
BRONX COMMUNITY BOARD #4	BRONX COMMUNITY BOARD #	22
BRONX COMMUNITY BOARD #5	BRONX COMMUNITY BOARD #	21
BRONX COMMUNITY BOARD #6	BRONX COMMUNITY BOARD #	21
BRONX COMMUNITY BOARD #7	BRONX COMMUNITY BOARD #	24
BRONX COMMUNITY BOARD #8	BRONX COMMUNITY BOARD #	26
BRONX COMMUNITY BOARD #9	BRONX COMMUNITY BOARD #	28

The value distribution of column, Agency Nam, and the correctness

4. Outliers

For the outlier, in the evaluation set, nearly 12% percentage data has been indicated as the outlier. Compared to the normal statistical percentile range method, which is 29.8%, the method that we used has decreased the number hugely but still giving several suggestions on the outlier selection.

VI. Conclusions

We select the data samples of 16412 records from the original dataset randomly based on the sample size calculator [21], with the confidence level as 99% and confidence interval to be 1. With the four techniques outlined of handling null values, outliers, invalid formats, and misspellings, the usability and reliability of the dataset has been significantly improved. The detailed evaluations above have demonstrated that the dataset has been cleaned properly and ready for use for data modeling or purpose of data processing.

This proposal provides multiple suggestions on the handling of data quality issues on Big data. We hope that this could be the initial guidance for processing on the data quality.

VII. References

1. Office of Payroll Administration. 2020. Citywide Payroll Data (Fiscal Year) https://data.cityofnewyork.us/City-Government/Citywide-Payroll-Data-Fiscal-Year-/k397-673e
2. Lianne & Justin. 2020. Data Cleaning in Python: the Ultimate Guide. https://towardsdatascience.com/data-cleaning-in-python-the-ultimate-guide-2020-c63b88bf0a0d
3. Kinha, Y. 2019. How to deal with missing values in your dataset. https://www.kdnuggets.com/2020/06/missing-values-dataset.html
4. Abhay Shukla. 2019. Levenshtein Distance in https://medium.com/@silpara/levenshtein-distance-in-pyspark-cf9faaf0fe0d
5. Zanid Haytam. 2019. OUTLIERS DETECTION IN PYSPARK #2 – INTERQUARTILE https://blog.zhaytam.com/2019/07/15/outliers-detection-in-pyspark-2-interquartile-range/
6. Zanid Haytam. 2019. OUTLIERS DETECTION IN PYSPARK #3 – https://blog.zhaytam.com/2019/08/06/outliers-detection-in-pyspark-3-k-means/
7. Khyati Mahendru. 2019. How to Determine the Optimal K for K-Means? https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb
8. Tryouge. 2017. Data-Cleaning-Library-Pyspark/getOutliers.py (github) https://github.com/tryouge/Data-Cleaning-Library-Pyspark/blob/master/getOutliers.py
9. Courtney Taylor. 2018. What Is the Interquartile Range Rule? https://www.thoughtco.com/what-is-the-interquartile-range-rule-3126244
10. Muhammad Ryan. 2018. How to Cluster and Detect Outlier at The Same Time https://medium.com/datadriveninvestor/how-to-clustering-and-detect-outlier-at-the-same-time-30576acd75d0
11. Agasti Kishor Dukare. 2020. Anomaly Detection in Python with Gaussian Mixture Models https://towardsdatascience.com/understanding-anomaly-detection-in-python-using-gaussian-mixture-model-e26e5d06094b
12. Okiriza Wibisono. 2016. What are the advantages to using a Gaussian Mixture Model clustering algorithm?

https://www.quora.com/What-are-the-advantages-to-using-a-Gaussian-Mixture-Model-clustering-algorithm

13. Weiqian Hou. 2015. K-means vs. GMM & PLSA
    http://www.math.sjsu.edu/~gchen/Math285F15/MATH285_Project_Report_Weiqian.pdf
14. Pavel Stefanovic, Olga Kurasova and Rokas Štrimaitis. 2019. The N-Grams Based Text Similarity Detection Approach Using Self-Organizing Maps and Similarity Measures
    https://www.mdpi.com/2076-3417/9/9/1870/pdf
15. Selva Prabhakaran. Cosine Similarity – Understanding the math and how it works
    https://www.machinelearningplus.com/nlp/cosine-similarity/
16. AHMED FAWZY GAD. 2020. Measuring Text Similarity Using the Levenshtein Distance
    https://blog.paperspace.com/measuring-text-similarity-using-levenshtein-distance/
17. HOW TO CLEAN DATA USING REGULAR EXPRESSIONS IN DATA QUALITY SERVICES?
    https://insightextractor.com/2012/12/19/how-to-clean-records-using-regular-expressions-in-data-quality-services/
18. Regular expression wikipedia
    https://en.wikipedia.org/wiki/Regular_expression
19. Pyspark: filter dataframe by regex with string formatting?
    https://stackoverrun.com/cn/q/12511042
20. Pyspark Dataframe creates new column based on function return value.
    https://stackoverflow.com/questions/53422473/pyspark-dataframe-create-new-column-based-on-function-return-value
21. Creative Research Systems.
    https://www.surveysystem.com/sscalc.htm
22. Python Regex (Regular Expressions) for Data Scientists
    https://www.dataquest.io/blog/regular-expressions-data-scientists/